1996

NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

MARSHALL SPACE FLIGHT CENTER
THE UNIVERSITY OF ALABAMA

DEVELOPMENT OF A COMPUTER ARCHITECTURE TO SUPPORT THE OPTICAL PLUME ANOMALY DETECTION (OPAD) SYSTEM

Prepared by : Constantine Katsinis, Ph.D.
Academic Rank: Associate Professor
Institution and Department: University of Alabama in Huntsville
Department of Electrical and Computer Engineering

NASA/MSFC:

Laboratory: Astrionics Laboratory
Division: Instrumentation and Control Division
Branch: Instrumentation Branch

MSFC Colleague: Anita Cooper

INTRODUCTION

The NASA OPAD spectrometer system relies heavily on extensive software which repetitively extracts spectral information from the engine plume and reports the amounts of metals which are present in the plume. The development of this software is at a sufficiently advanced stage where it can be used in actual engine tests to provide valuable data on engine operation and health. This activity will continue and, in addition, the OPAD system is planned to be used in flight aboard space vehicles. The two implementations, test-stand and in-flight, may have some differing requirements. For example, the data stored during a test-stand experiment are much more extensive than in the in-flight case. In both cases though, the majority of the requirements are similar. New data from the spectrograph is generated at a rate of once every 0.5 sec or faster. All processing must be completed within this period of time to maintain real-time performance.

Every 0.5 sec, the OPAD system must report the amounts of specific metals within the engine plume, given the spectral data. At present, the software in the OPAD system performs this function by solving the inverse problem. It uses powerful physics-based computational models (the SPECTRA code), which receive amounts of metals as inputs to produce the spectral data that would have been observed, had the same metal amounts been present in the engine plume. During the experiment, for every spectrum that is observed, an initial approximation is performed using neural networks to establish an initial metal composition which approximates as accurately as possible the real one. Then, using optimization techniques, the SPECTRA code is repetitively used to produce a fit to the data, by adjusting the metal input amounts until the produced spectrum matches the observed one to within a given level of tolerance. This iterative solution to the original problem of determining the metal composition in the plume requires a relatively long period of time to execute the software in a modern single-processor workstation, and therefore real-time operation is currently not possible.

A different number of iterations may be required to perform spectral data fitting per spectral sample. Yet, the OPAD system must be designed to maintain real-time performance in all cases. Although faster single-processor workstations are available for execution of the fitting and SPECTRA software, this option is unattractive due to the excessive cost associated with very fast workstations and also due to the fact that such hardware is not easily expandable to accommodate future versions of the software which may require more processing power.

Initial research has already demonstrated that the OPAD software can take advantage of a parallel computer architecture to achieve the necessary speedup. Current work has improved the software by converting it into a form which is easily parallelizable. Timing experiments have been performed to establish the computational complexity and execution speed of major components of the software. This work provides the foundation of future work which will create a fully parallel version of the software executing in a shared-memory multiprocessor system.

OPAD SOFTWARE PARALLELIZATION

The OPAD software consists of three major components: 1) The SPECTRA code which receives an array specifying the metal composition in the plume and uses physics-based models to produce the spectrum that would have been observed had the specified metals been in the plume,

2) the neural network code which receives an observed spectrum and produces estimates of the metal composition in the plume, and 3) the optimization, or fitting, code which uses gradient techniques to match the estimated and observed spectra. During normal operation, the OPAD system repetitively acquires the spectrum from the plume, uses the neural network code to produce an initial approximation to the metal composition, and then, under control of the optimization code, repetitively calls the SPECTRA code to produce successive approximate spectra, which are increasingly better (closer) to the observed spectrum until the difference between observed and estimated spectra becomes smaller than a specific limit.

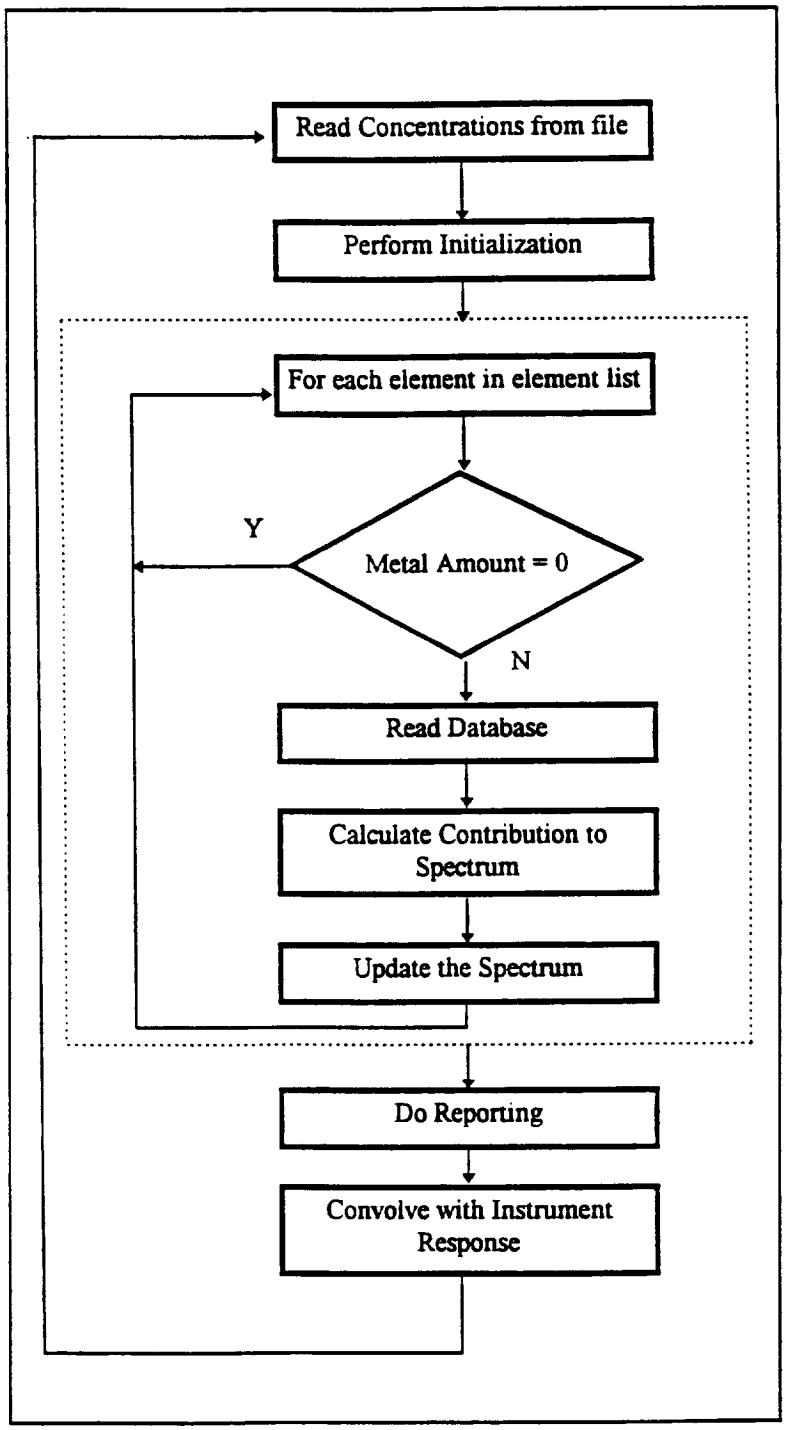

Figure 1. Original program flow

Figure 1 shows the program flow of the SPECTRA code component. During real-time operation, metal concentrations are received from the neural network code rather than being read from a file. Although they are quite extensive and requiring some non-trivial computational resources to train and set-up, neither the spectral code nor the optimization code appear to be computationally demanding during real-time operation. The step which requires a relatively significant amount of time to execute is the one where the contribution of each individual metal to the spectrum is calculated. Table 1 shows the time required to execute that step on a workstation (of modest capabilities) for each of the nineteen metals which are currently of interest. All numbers are in milliseconds. It is apparent from the table that the execution time of that step varies significantly from negligible (less than the time resolution) for two metals to more than three seconds for two other ones. Also, the step where the convolution of the resulting spectrum with the

instrument response takes place requires some non-negligible amount of time. The total processing time, if all steps are performed serially, is approximately 14 seconds. If the processing for each metal were allocated to a different processor then all contributions to the spectrum would be calculated within approximately 3.8 seconds. Then, nineteen processors, each used for the above experiment, would be sufficient to execute the SPECTRA code once in 0.5 seconds. Of course, the above processing allocation results in many processors being idle for a significant amount of time. By optimizing the SPECTRA code so that the original 14 seconds of processing are distributed evenly over the nineteen processors, the total processing time reduces to less than 0.1 second, assuming processors 8 times faster that the workstation.

Figure 2 shows the flowchart of the SPECTRA code after its conversion to parallel form. It assumes that N+1 processors are available, with one processor performing initialization and coordination and N processors performing the real work. More than one can be allocated to the same processor, the allocation being such that all N processors receive approximately the same amount of work so that idle time is minimal.

TABLE 1. Processing Times per Element (msec)

copper	136	sodium	29
calcium	45	potassium	0
magnesium	29	lithium	15
molybdenum	3300	silver	32
aluminum	0	hafnium	1045
lead	73	scandium	45
titanium	336	vanadium	933
chromium	153	manganese	1717
iron	3787	cobalt	1239
nickel	1175		
convolution	139		

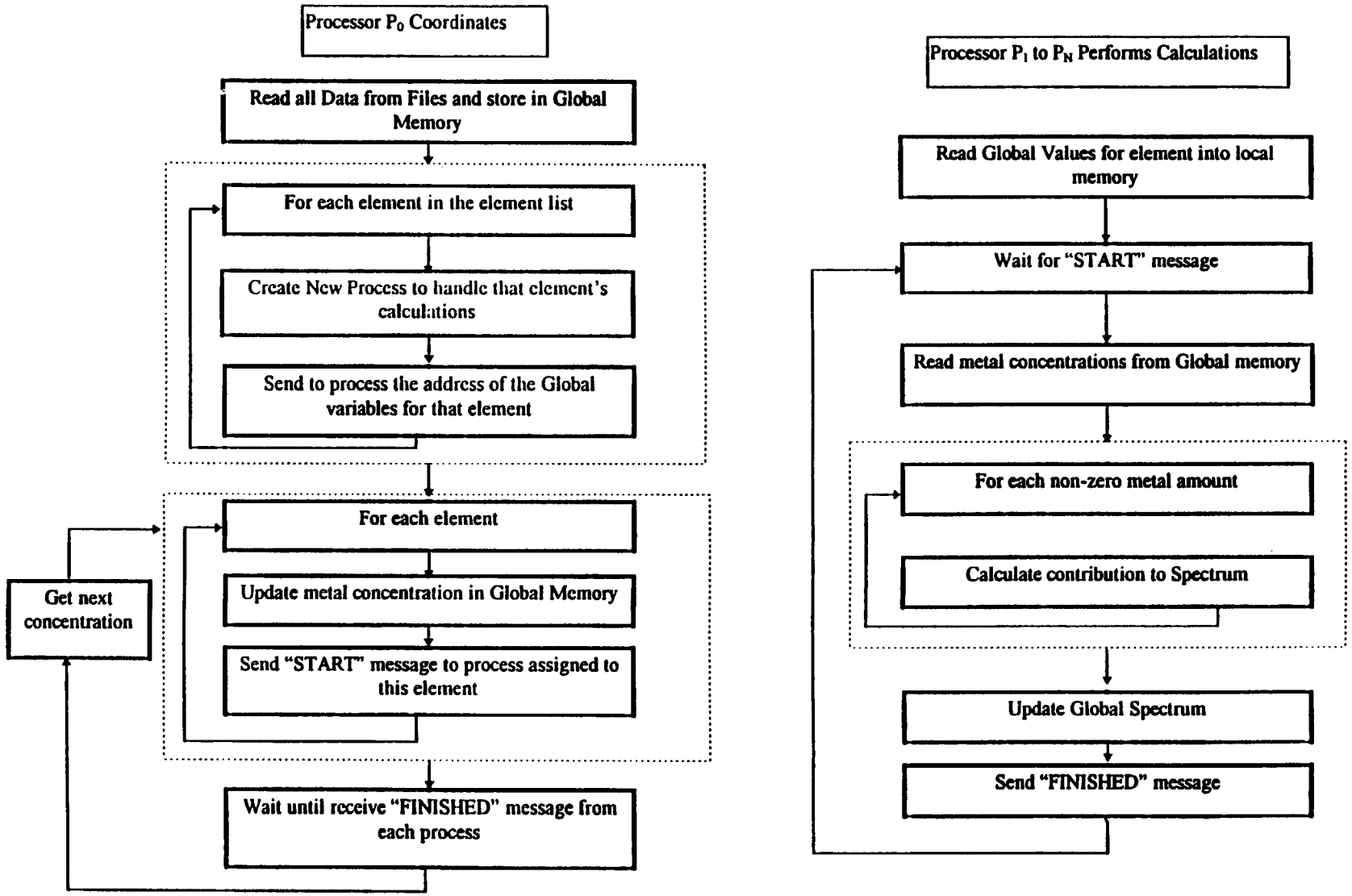

Figure 2. Parallel program flow